

# VB.NET - OPERATORS

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. VB.Net is rich in built-in operators and provides following type of commonly used operators:

- Arithmetic Operators
- Comparison Operators
- Logical/Bitwise Operators
- Bit Shift Operators
- Assignment Operators
- Miscellaneous Operators

This tutorial will explain the most commonly used operators.

## Arithmetic Operators

Following table shows all the arithmetic operators supported by VB.Net. Assume variable **A** holds 2 and variable **B** holds 7 then:

[Show Examples](#)

Operator	Description	Example
$\wedge$	Raises one operand to the power of another	$B^A$ will give 49
+	Adds two operands	$A + B$ will give 9
-	Subtracts second operand from the first	$A - B$ will give -5
*	Multiply both operands	$A * B$ will give 14
/	Divide one operand by another and returns a floating point result	$B / A$ will give 3.5
$\backslash$	Divide one operand by another and returns an integer result	$B \backslash A$ will give 3
MOD	Modulus Operator and remainder of after an integer division	$B \text{ MOD } A$ will give 1

## Comparison Operators

Following table shows all the comparison operators supported by VB.Net. Assume variable **A** holds 10 and variable **B** holds 20 then:

[Show Examples](#)

Operator	Description	Example
----------	-------------	---------

==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true.
<>	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A <> B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Apart from the above, VB.Net provides three more comparison operators, which we will be using in forthcoming chapters; however, we give a brief description here.

- **Is Operator** - It compares two object reference variables and determines if two object references refer to the same object without performing value comparisons. If object1 and object2 both refer to the exact same object instance, result is **True**; otherwise, result is False.
- **IsNot Operator** - It also compares two object reference variables and determines if two object references refer to different objects. If object1 and object2 both refer to the exact same object instance, result is **False**; otherwise, result is True.
- **Like Operator** - It compares a string against a pattern.

## Logical/Bitwise Operators

Following table shows all the logical operators supported by VB.Net. Assume variable A holds Boolean value True and variable B holds Boolean value False then:

[Show Examples](#)

Operator	Description	Example
And	It is the logical as well as bitwise AND operator. If both the operands are true then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A And B) is False.
Or	It is the logical as well as bitwise OR operator. If any of the two operands is true then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A Or B) is True.
Not	It is the logical as well as bitwise NOT operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	Not(A And B) is True.

Xor	It is the logical as well as bitwise Logical Exclusive OR operator. It returns True if both expressions are True or both expressions are False; otherwise it returns False. This operator does not perform short-circuiting, it always evaluates both expressions and there is no short-circuiting counterpart of this operator	A Xor B is True.
AndAlso	It is the logical AND operator. It works only on Boolean data. It performs short-circuiting.	(A AndAlso B) is False.
OrElse	It is the logical OR operator. It works only on Boolean data. It performs short-circuiting.	(A OrElse B) is True.
IsFalse	It determines whether an expression is False.	
IsTrue	It determines whether an expression is True.	

## Bit Shift Operators

We have already discussed the bitwise operators. The bit shift operators perform the shift operations on binary values. Before coming into the bit shift operators, let us understand the bit operations.

Bitwise operators work on bits and perform bit by bit operation. The truth tables for  $\&$ ,  $|$ , and  $\wedge$  are as follows:

p	q	p & q	p   q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Assume if A = 60; and B = 13; Now in binary format they will be as follows:

A = 0011 1100

B = 0000 1101

-----

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

~A = 1100 0011

We have seen that the Bitwise operators supported by VB.Net are And, Or, Xor and Not. The Bit shift operators are  $\gg$  and  $\ll$  for left shift and right shift respectively.

Assume that the variable A holds 60 and variable B holds 13 then:

[Show Examples](#)

Operator	Description	Example
And	Bitwise AND Operator copies a bit to the result if it exists in both operands.	(A AND B) will give 12 which is 0000 1100
Or	Binary OR Operator copies a bit if it exists in either operand.	(A Or B) will give 61 which is 0011 1101
Xor	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A Xor B) will give 49 which is 0011 0001
Not	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(Not A ) will give -60 which is 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

## Assignment Operators

There are following assignment operators supported by VB.Net:

[Show Examples](#)

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand(floating point division)	C /= A is equivalent to C = C / A
\=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand (Integer division)	C \= A is equivalent to C = C \A
^=	Exponentiation and assignment operator. It raises the left operand to the power of the right operand and assigns the result to left operand.	C^=A is equivalent to C = C ^ A

<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Concatenates a String expression to a String variable or property and assigns the result to the variable or property.	Str1 &= Str2 is same as Str1 = Str1 & Str2

## Miscellaneous Operators

There are few other important operators supported by VB.Net.

[Show Examples](#)

Operator	Description	Example
AddressOf	Returns the address of a procedure.	<code>AddHandler Button1.Click, AddressOf Button1_Click</code>
Await	It is applied to an operand in an asynchronous method or lambda expression to suspend execution of the method until the awaited task completes.	<code>Dim result As res = Await AsyncMethodThatReturnsResult() Await AsyncMethod()</code>
GetType	It returns a Type object for the specified type. The Type object provides information about the type such as its properties, methods, and events.	<code>MsgBox(GetType(Integer).ToString())</code>
Function Expression	It declares the parameters and code that define a function lambda expression.	<code>Dim add5 = Function(num As Integer) num + 5 'prints 10 Console.WriteLine(add5(5))</code>
If	It uses short-circuit evaluation to conditionally return one of two values. The If operator can be called with three arguments or with two arguments.	<code>Dim num = 5 Console.WriteLine(If(num &gt;= 0, "Positive", "Negative"))</code>

## Operators Precedence in VB.Net

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:

For example  $x = 7 + 3 * 2$ ; Here x is assigned 13, not 20 because operator \* has higher precedence than + so it first get multiplied with  $3*2$  and then adds into 7.

Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

[Show Examples](#)

Operator	Precedence
Await	Highest
Exponentiation (^)	
Unary identity and negation (+, -)	
Multiplication and floating-point division (*, /)	
Integer division (\)	
Modulus arithmetic (Mod)	
Addition and subtraction (+, -)	
Arithmetic bit shift (<<, >>)	
All comparison operators (=, <>, <, <=, >, >=, Is, IsNot, Like, TypeOf...Is)	
Negation (Not)	
Conjunction (And, AndAlso)	
Inclusive disjunction (Or, OrElse)	
Exclusive disjunction (Xor)	Lowest